This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

THIS PAGE BLANK (USPTO)

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G05B 13/04	A1	(11) International Publication Number: WO 00/41045 (43) International Publication Date: 13 July 2000 (13.07.00)

(21) International Application Number: PCT/US99/27726

(22) International Filing Date: 22 November 1999 (22.11.99)

(30) Priority Data:
09/224,439 31 December 1998 (31.12.98) US

(71) Applicant: HONEYWELL INC. [US/US]; Honeywell Plaza, Minneapolis, MN 55408 (US).

(72) Inventor: LU, Z., Joseph; 19513 N. 73rd Lane, Glendale, AZ 85308 (US).

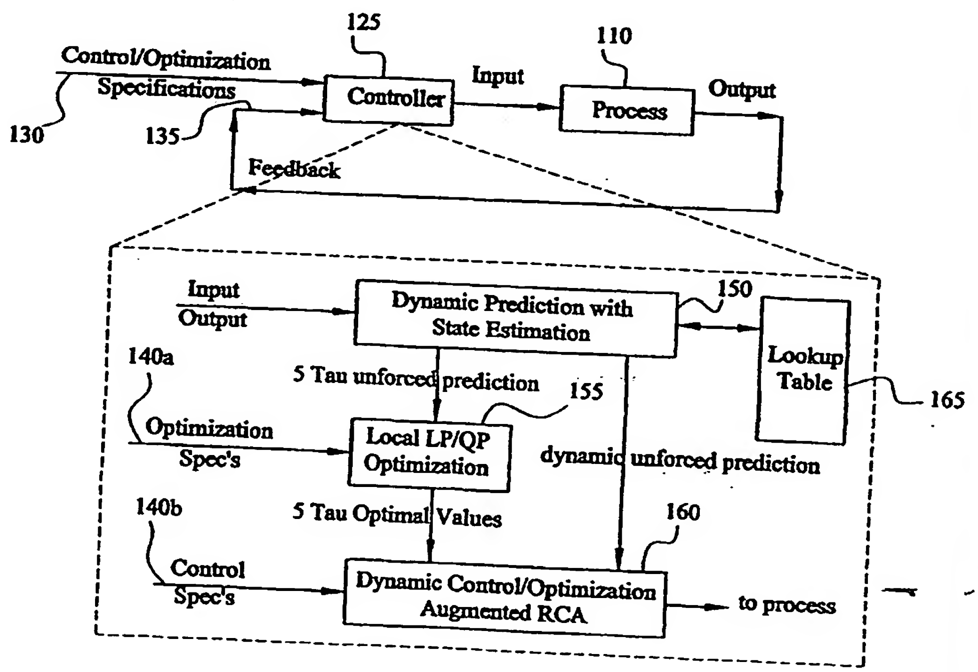
(74) Agent: MIOLOGOS, Anthony; Honeywell Inc., Honeywell Plaza MN12-8251, P.O. Box 524, Minneapolis, MN 55440-0524 (US).

(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEMS FOR GENERATING AND USING A LOOKUP TABLE WITH PROCESS FACILITY CONTROL SYSTEMS AND MODELS OF THE SAME, AND METHODS OF OPERATING SUCH SYSTEMS

(57) Abstract



Systems and methods of operating the same are introduced for populating and using lookup tables with process facility control systems and models of the same. An exemplary computer system for use with a process facility having a plurality of associated processes, and includes both a memory and a processor. The memory is capable of maintaining (i) a data structure having a plurality of accessible fields and (ii) a model of at least a portion of the plurality of associated processes. The model may include a mathematical representation of at least a portion of the at least one process, defining certain relationships among inputs and outputs of the at least one process. The processor is capable of populating ones of the plurality of accessible fields of the data structure using the model iteratively with a range of possible values of the at least one measurable characteristic. The computer system is capable of using the range of possible values of the at least one measurable characteristic to predict an unforced response associated with the at least one process.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

SYSTEMS FOR GENERATING AND USING A LOOKUP TABLE WITH PROCESS FACILITY CONTROL SYSTEMS AND MODELS OF THE SAME, AND METHODS OF OPERATING SUCH SYSTEMS

COPYRIGHT NOTICE

A portion of the disclosure of this patent document (software listings in APPENDICES A and B) contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of this patent document or the patent disclosure, as it appears in the United States Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights and protection whatsoever.

CROSS-REFERENCE TO RELATED PATENT DOCUMENTS

The present invention is related to that disclosed in (i) United States Patent No. 5,351,184 entitled "METHOD OF MULTIVARIABLE PREDICTIVE CONTROL UTILIZING RANGE CONTROL;" (ii) United States Patent No. 5,561,599 entitled "METHOD OF INCORPORATING INDEPENDENT FEEDFORWARD CONTROL IN A MULTIVARIABLE PREDICTIVE CONTROLLER;" (iii) United States Patent No. 5,574,638 entitled "METHOD OF OPTIMAL SCALING OF VARIABLES IN A MULTIVARIABLE PREDICTIVE CONTROLLER UTILIZING RANGE CONTROL;" (iv) United States Patent No. 5,572,420 entitled "METHOD OF OPTIMAL CONTROLLER DESIGN OF MULTIVARIABLE PREDICTIVE CONTROL UTILIZING RANGE CONTROL" (the "'420 Patent"); (v) United States Patent No. 5,758,047 entitled "METHOD OF PROCESS CONTROLLER OPTIMIZATION IN A MULTIVARIABLE PREDICTIVE CONTROLLER;" (vi) United States Patent Application Serial No. 08/490,499, filed on June 14, 1995, entitled "Method of Process Controller Optimization in a Multivariable Predictive Controller;" (vii) United States Patent Application Serial No. 08\850,288 entitled "SYSTEMS AND METHODS FOR GLOBALLY OPTIMIZING A PROCESS FACILITY;" (viii) United States Patent Application Serial No. 08\851,590 entitled "SYSTEMS FOR GENERATING AND USING A LOOKUP TABLE WITH PROCESS FACILITY CONTROL SYSTEMS AND MODELS OF THE SAME, AND METHODS OF OPERATING SUCH SYSTEMS;" and (ix) United States Patent Application Serial No. 09\137,358 entitled "CONTROLLERS THAT DETERMINE OPTIMAL TUNING PARAMETERS FOR USE IN PROCESS CONTROL SYSTEMS AND METHODS OF OPERATING THE SAME;" and (x) United States Patent Application Serial No. (Attorney Docket No. I20 25206),

entitled "PROCESS FACILITY CONTROL SYSTEMS USING AN EFFICIENT PREDICTION FORM AND METHODS OF OPERATING THE SAME" (which application is filed concurrently herewith), all of which are commonly assigned to the assignee of the present invention. The disclosures of these related patent applications are incorporated herein by reference for all purposes as if fully set forth herein

TECHNICAL FIELD OF THE INVENTION

The present invention is directed, in general, to control systems for process facilities and, more specifically, to systems for generating and using lookup tables with process facility control systems and models of the same, and methods of operating such systems, all for use to optimize process facilities.

BACKGROUND OF THE INVENTION

Presently, process facilities (*e.g.*, a manufacturing plant, a mineral or crude oil refinery, etc.) are managed using distributed control systems. Contemporary control systems include numerous modules tailored to control or monitor various associated processes of the facility. Conventional means link these modules together to produce the distributed nature of the control system. This affords increased performance and a capability to expand or reduce the control system to satisfy changing facility needs.

Process facility management providers, such as HONEYWELL, INC., develop control systems that can be tailored to satisfy wide ranges of process requirements (*e.g.*, global, local or otherwise) and facility types (*e.g.*, manufacturing, refining, etc.). A primary objective of such providers is to centralize control of as many processes as possible to improve an overall efficiency of the facility. Each process, or group of associated processes, has certain input (*e.g.*, flow, feed, power, etc.) and output (*e.g.*, temperature, pressure, etc.) characteristics associated with it.

In recent years, model predictive control ("MPC") techniques have been used to optimize certain processes as a function of such characteristics. One technique uses algorithmic representations to estimate characteristic values (represented as parameters, variables, etc.) associated with them that can be used to better control such processes. In recent years, physical, economic and other factors have been incorporated into control systems for these associated processes. Examples of such techniques are described in United States Patent No. 5,351,184 entitled "METHOD OF MULTIVARIABLE PREDICTIVE CONTROL UTILIZING RANGE CONTROL;" United States Patent No.

5,561,599 entitled "METHOD OF INCORPORATING INDEPENDENT FEEDFORWARD CONTROL IN A MULTIVARIABLE PREDICTIVE CONTROLLER;" United States Patent No. 5,574,638 entitled "METHOD OF OPTIMAL SCALING OF VARIABLES IN A MULTIVARIABLE PREDICTIVE CONTROLLER UTILIZING RANGE CONTROL;" United States Patent No. 5,572,420 entitled "METHOD OF OPTIMAL CONTROLLER DESIGN OF MULTIVARIABLE PREDICTIVE CONTROL UTILIZING RANGE CONTROL" (the "'420 Patent"); United States Patent Application Serial No. 08\850,288 entitled "SYSTEMS AND METHODS FOR GLOBALLY OPTIMIZING A PROCESS FACILITY;" United States Patent Application Serial No. 08\851,590 entitled "SYSTEMS AND METHODS USING BRIDGE MODELS TO GLOBALLY OPTIMIZE A PROCESS FACILITY;" and United States Patent Application Serial No. 09\137,358 entitled "CONTROLLERS THAT DETERMINE OPTIMAL TUNING PARAMETERS FOR USE IN PROCESS CONTROL SYSTEMS AND METHODS OF OPERATING THE SAME," all of which are commonly owned by the assignee of the present invention and incorporated herein above by reference for all purposes.

Generally speaking, one problem is that conventional efforts, when applied to specific processes, tend to be non-cooperative (*e.g.*, non-global, non-facility wide, etc.) and may, and all too often do, detrimentally impact the efficiency of the process facility as a whole. For instance, many MPC techniques control process variables to predetermined set points. Oftentimes the set points are a best estimate of a value of the set point or set points. When a process is being controlled to a set point, the controller may not be able to achieve the best control performances, especially under process/model mismatch.

To further enhance the overall performance of a control system, it is desirable to design a controller that deals explicitly with plant or model uncertainty. The '420 Patent, for example, teaches methods of designing a controller utilizing range control. The controller is designed to control a "worst case" process. An optimal controller for the process is achieved and, if the actual process is not a "worst case process," the performance of the controller is better than anticipated.

There are a number of well known PID "tuning" formulas, or techniques, and the most common, or basic, PID algorithm includes three known user specified tuning parameters (K, $_{1}$, $_{2}$) whose values determine how the controller will behave. These parameters are determined either by trial and error or through approaches that require knowledge of the process. Although many of these approaches, which are commonly algorithms, have provided improved control, PID controller performance tuned by such

algorithms usually degrades as process conditions change, requiring a process engineer, or operator, to monitor controller performance. If controller performance deteriorates, the process engineer is required to "re-tune" the controller.

Controller performance deteriorates for many reasons, although the most common cause is changing dynamics of the process. Since PID controller performance has been related to the accuracy of the process model chosen, a need exists for PID controllers that allows for such uncertainty by accounting for changing system dynamics. Further, the requirement for ever-higher performance control systems demands that system hardware maximize software performance. Conventional control system architectures are made up of three primary components: (i) a processor, (ii) a system memory and (iii) one or more input/output devices. The processor controls the system memory and the input/output ("I/O") devices. The system memory stores not only data, but also instructions that the processor is capable of retrieving and executing to cause the control system to perform one or more desired functions. The I/O devices are operative to interact with an operator through a graphical user interface, and with the facility as a whole through a network portal device and a process interface.

Over the years, the quest for ever-increasing process control system speeds has followed different directions. One approach to improve control system performance is to increase the rate of the clock that drives the system hardware. As the clock rate increases, however, the system hardware's power consumption and temperature also increase. Increased power consumption is expensive and high circuit temperatures may damage the process control system. Further, system hardware clock rate may not increase beyond a threshold physical speed at which signals may be processed. More simply stated, there is a practical maximum to the clock rate that is acceptable to conventional system hardware.

An alternate approach to improve process control system performance is to increase the number of instructions executed per clock cycle by the system processor ("processor throughput"). One technique for increasing processor throughput calls for the processor to be divided into separate processing stages. Instructions are processed in an "assembly line" fashion in the processing stages. Each processing stage is optimized to perform a particular processing function, thereby causing the processor as a whole to become faster. There is again a practical maximum to the clock rate that is acceptable to conventional system hardware.

Since there are discernable physical limitations to which conventional system

hardware may be utilized, a need exists broadly for an approach that decreases the number of instructions required to preform the functions of the process control system. A need exists for such an approach that accounts for process uncertainty by accounting for changing process dynamics.

SUMMARY OF THE INVENTION

To address the above-discussed deficiencies of the prior art, it is a primary object of the present invention to provide systems and methods of operating such systems for populating and using lookup tables with process facility control systems, as well as models of the same. In accordance with an exemplary embodiment below-discussed, the principles of the present invention may be used to define and populate a lookup table in response to the needs of a global controller. The lookup table is populated with a range of possible values of at least one measurable characteristic associated with one or more processes of the process facility and in accordance with a model of at least a portion of the same.

Rather than calculate and re-calculate certain characteristics associated with a process or process model, which would consume significant system resources, the present invention introduces a data structure capable of maintaining a range of possible values of one or more of such certain characteristics. Use of the lookup table in lieu of execution and re-execution of the instructions for performing characteristic calculations decreases the number of instructions required to preform the functions of the process control system. The lookup table, once suitably populated, accounts for process uncertainty by maintaining the range of possible values, thereby accounting for changing process dynamics.

An exemplary computer system for use with a process facility that is capable of populating a data structure in accordance with the principles of the present invention includes both a memory and a processor. The memory is capable of maintaining (i) the data structure, which has a plurality of accessible fields, and (ii) a model of at least a portion of at least one process of a plurality of associated processes of the process facility. The model may advantageously include a mathematical representation of at least a portion of the at least one process, defining certain relationships among inputs and outputs of the at least one process. The processor is capable of populating ones of the plurality of accessible fields of the data structure using the model iteratively with a range of possible values of the at least one measurable characteristic. The computer

system is capable of using the range of possible values of the at least one measurable characteristic to predict an unforced response associated with the at least one process.

In accordance with an important aspect hereof, the data structure may be populated and maintained on-line (*e.g.*, at a controller, distributed through a process control system, etc.), off-line (*e.g.*, standalone computer, computer network, etc.), or through some suitable combination of the same. Likewise, the data structure may remain static upon population, be dynamic, or be modifiable, at least in part.

Those skilled in the art will understand that "controllers" may be implemented in hardware, software, or firmware, or some suitable combination of the same, and, in general, that the use of computing systems in control systems for process facilities is known. The phrase "associated with" and derivatives thereof, as used herein, may mean to include, be included within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, be a property of, be bound to or with, have, have a property of, or the like; the term "include" and derivatives thereof, as used herein, are defined broadly, meaning inclusion without limitation; and the term "or," as used herein, means and/or.

The foregoing has outlined rather broadly the features and technical advantages of the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features and advantages of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art should appreciate that they may readily use the conception and the specific embodiment disclosed as a basis for modifying or designing other structures for carrying out the same purposes of the present invention. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the invention in its broadest form.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like numbers designate like objects, and in which:

FIGURE 1a illustrates a simple block diagram of an exemplary process facility with which the present invention may be used;

FIGURE 1b illustrates a detailed block diagram of one of the exemplary local controllers introduced in FIGURE 1a;

FIGURE 2 illustrates a flow diagram of an exemplary method for populating a data structure in accordance with the principles of the present invention; and

FIGURE 3 illustrates an exemplary two-dimensional graphical representation of MV and PV curves in accordance with the principles of the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

In accordance with the above-given summary, computer systems, and methods of operating the same, are introduced herein for populating and using lookup tables with process facility control systems, as well as models of the same. Before undertaking a detailed description of an advantageous embodiment of the present invention, and discussing the various benefits and aspects of the same, it is useful to understand conceptually the operation and control structure of an exemplary process facility.

Initial reference is therefore made to FIGURE 1a, wherein a simple block diagram of such a process facility (generally designated 100) is illustrated. Exemplary process facility 100 is operative to process raw materials, and includes a control center 105, six associated processes 110a to 110f that are arranged into three stages and a control system (generally designated 115). The term "include," as well as derivatives thereof, as used throughout this patent document, is defined broadly to mean inclusion without limitation.

Exemplary control center 105 illustrates a central area that is commonly operator manned (not shown) for centrally monitoring and for centrally controlling the three exemplary process stages. A first process stage includes three raw material grinders 110a to 110c that operate to receive a "feed" of raw material core and to grind the same, such as using a pulverizer or grinding wheel, into smaller particles of raw material. The term "or," as it is used throughout this patent document, is inclusive, meaning and/or. The second process stage includes a washer 110d that operates to receive the ground raw materials and clean the same to remove residue from the first stage. The third process stage includes a pair of separators 110e and 110f that operate to receive the ground and washed raw materials and separate the same, such as into desired minerals and any remaining raw materials. As this process facility is provided for illustrative purposes only and the principles of such are known, further discussion of the same is beyond the scope of this patent document.

Exemplary control system 115 illustratively includes a global controller 120 and six local controllers 125a to 125f, each of which is implemented in software and

executable by a suitable conventional computer system (*e.g.*, standalone, network, etc.), such as any of HONEYWELL, INC.'s AM K2LCN, AM K4LCN, AM HMPU, AxM or like systems. Those skilled in the art will understand that such controllers may be implemented in hardware, software, or firmware, or some suitable combination of the same; in general, the use of computing systems in control systems for process facilities is known.

Global controller 120 is associated with each of local controllers 125, directly or indirectly, to allow communication of information between the same. The phrase "associated with" and derivatives thereof, as used throughout this patent document, may mean to include within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, be a property of, be bound to or with, have, have a property of, or the like.

Global controller 120 monitors measurable characteristics (*e.g.*, status, temperature, utilization, efficiency, cost and other economic factors, etc.) of associated processes 110, either directly or indirectly (as shown, through local controllers 125 associated with processes 110). Depending upon the implementation, such monitoring may be of an individual process, group of processes, the facility as a whole, or otherwise. Similarly, local controllers 125 monitor associated processes 110, and, more particularly, monitor certain characteristics of associated processes 110.

Global controller 120 generates, in response to such monitoring efforts, control data that may be communicated via local controllers 125 to associated processes 110 to optimize process facility 100. The phrase "control data," as used herein, is defined as any numeric, qualitative or other value generated by global controller 120 to globally control (*e.g.*, direct, manage, modify, recommend to, regulate, suggest to, supervise, cooperate, etc.) a particular process, a group of processes, a facility, a process stage, a group of process stages, a sequence of processes or process stages, or the like to optimize the facility. Local controllers 125 operate to varying degrees in accordance with the control data to control the associated processes, and, more particularly, to modify one or more processes and improve the monitored characteristics and the facility.

According to an advantageous embodiment, the control data may be dynamically generated using a lookup table defined and populated in accordance with the principles hereof, and such control data generation is based, at least in part, upon a given facility's efficiency, production or economic cost, and, most preferably, all three.

The lookup table may be populated and maintained on-line (*e.g.*, at global controller 120, at local controller 125, distributed within control system 115, etc.), off-line (*e.g.*, standalone computer, network computer, etc.), or through some suitable combination of the same; likewise, the lookup table may be static upon population, be dynamic, or be modifiable, at least in part.

The global controller 120 and the local controllers 125 may suitably use one or more such lookup tables to control processes 110 to conserve processing resources and increase the overall speed of control system 115. Control system 115 achieves a high level of both global and local monitoring, and cooperative control of associated processes 110 among controllers 120 and 125, by allowing the local controllers 125 to vary their individual or respective compliance with the control data. Varying degrees of compliance by local controllers 125 may range between full compliance and noncompliance. The relationship between global controller 120 and various ones of local controllers 110 may be master-slave (full compliance), cooperative (varying compliance, *e.g.*, using control data as a factor in controlling the associated processes), complete disregard (noncompliance), as well as anywhere along that range.

Depending upon the implementation and needs of a given facility, the relationship between global controller 120 and specific local controllers 125 may be static (*i.e.*, always only one of compliance, cooperative, or noncompliance), dynamic (*i.e.*, varying over time, such as within a range between compliance and noncompliance, some lesser range therebetween, or otherwise), or varying between the same. One or more specific processes 110, and facility 100 as a whole, may be dynamically and cooperatively controlled as a function of local and global optimization efforts, and such dynamic and cooperative control is independent of the relationship between global controller 120 and specific local controllers 125, as described above.

Turning to FIGURE 1b, illustrated is a more detailed block diagram of one of the exemplary local controllers 125 that is associated with one or group of associated processes 110. Local controller 125 uses a single loop model predictive control ("SL-MPC") structure that uses an efficient matrix prediction form in accordance with the principles of the present invention, as well as an analytical control solution map to reduce utilization of processing resources relative to conventional MPC technology.

According to the illustrated embodiment, local controller 125 receives as inputs, control/optimization specifications 130 (*e.g.*, bounds, ranges, tolerances, control points, etc.) and feedback data 135 (*e.g.*, output of associated process 110).

Control/optimization specifications 130 may be received from any of a number of sources depending upon the associated process or group of associated processes 110, an associated process facility or any other factor. For example, any of control/optimization specifications 130 may be received from an operator of a control center for the associated process facility, retrieved from a database or data repository, received from another associated controller (*e.g.,* one or more local controllers 125, global controller 120, or a suitable combination thereof), etc.

Control/optimization specifications 130 include two types of variables: (1) a first variable ("MV") that may be manipulated, such as flow, feed, air blower, etc; and (2) a second variable ("DV") that cannot be manipulated and is a disturbance variable, such as burn rate, fuel quality per unit, etc. Feedback data 135 is a third variable ("CV") that is responsive to MVs and DVs, and is an output of associated process 110, such as pressure, temperature, etc. A sub-variable ("PV") of Feedback data 135 is indicative of the iterative response of the associated process 110 to monitoring and control by the local controller 125. Many, if not all, of such MVs, DVs and CVs represent measurable characteristics of associated process 110 that may be suitably monitored by local controller 125.

Local controller 125 includes a dynamic prediction task with state estimation 150, a local linear program/quadratic program ("LP/QP") optimization task 155, a dynamic control/optimization augmented range control algorithm ("RCA") 160 and a lookup table 165. Exemplary dynamic prediction task 150 receives CVs and operates to generate an array of multiple predictions (or dynamic unforced predictions) and, at 5 tau (response time close to end), an unforced prediction for values associated with associated process 110. The CVs represent feedback data 135 (*e.g.*, inputs, outputs, etc.) associated with process 105, and dynamic prediction task 150 operates to accesses lookup table 165 and selects one or more values from the range of possible values, such selection being responsive, at least in part, to the received feedback data 135. A preferred method of using data structures, such as lookup table 165, or functionally equivalent dedicated circuitry, to maintain a range of possible values for one or more measurable characteristics associated with a process is disclosed and described in United States Patent Application Serial No. (Attorney Docket No. I20 25207), entitled "PROCESS FACILITY CONTROL SYSTEMS USING AN EFFICIENT PREDICTION FORM AND METHODS OF OPERATING THE SAME" and filed concurrently herewith, the disclosure of which has previously been incorporated herein by reference for all purposes as if fully

set forth herein.

Exemplary local LP/QP optimization task 155 receives optimization specifications 140a and, in response to the unforced prediction, operates to generate, at 5 tau, optimal values associated with associated process 110.

A preferred method of performing the foregoing task is disclosed and described in United States Patent No. 5,758,047, entitled "METHOD OF PROCESS CONTROLLER OPTIMIZATION IN A MULTIVARIABLE PREDICTIVE CONTROLLER," which is commonly owned by the assignee of this patent document and related invention, the disclosure of which has previously been incorporated herein by reference for all purposes as if fully set forth herein. Most preferably, optimization specifications 140a are associated, directly or indirectly, with an economic value of the output of associated process 110. According to an advantageous embodiment, the unforced prediction may suitably be represented as a single variable and the LP/QP optimization task may be a linear determination of a minimum value or a maximum value, or a quadratic determination of a desired value. Exemplary dynamic control/optimization augmented RCA 160 receives control specifications 140b and, in response to receiving the array of multiple predictions (from dynamic prediction task 150) and the optimal values (from local LP/QP optimization task 155), operates to generate control values, the MVs, that are input to associated process 110. An important aspect of exemplary local controller 125 is the use of control/optimization specifications 140 and feedback data 135 to locally unify economic/operational optimization with MPC dynamically for a specific process or group of processes.

Note the distinction between the foregoing discussion which introduces a very powerful multi-loop MPC embodiment having a well defined and dynamic interaction/interleaving relation among global and local controllers and the single loop controller embodiment described in United States Patent Application Serial No. (Attorney Docket No. I20 25207), the disclosure of which has previously been incorporated herein by reference for all purposes. Those skilled in the art will understand the relationship among these embodiments and the applicability of the principles of the present invention.

Turning now to FIGURE 2, illustrated is a flow diagram of an exemplary method (generally designated 200) for populating a data structure 165, shown as a lookup table, in accordance with the principles of the present invention (this discussion of FIGURE 2 makes concurrent reference to FIGURES 1a and 1b). The phrase "data

structure," as the same is used herein, is defined broadly as any syntactic structure of expressions, data or other values or indicia, including both logical and physical structures. A data structure may therefore be any array (*i.e.*, any arrangement of objects into one or more dimensions, *e.g.*, a matrix, a table, etc.), or other like grouping, organization, or categorization of objects in accordance herewith.

For purposes of illustration, a processor 205 and a memory 210 are introduced. Exemplary memory 210 is operative to store, or to maintain, lookup table 165, along with the various tasks/ instructions (generally designated 215) comprising method 200. Exemplary processor 205 is operative to select and execute tasks/ instructions 215 which, in turn, cause processor 205 to perform the functions of method 200.

To begin, processor 205 is directed through the execution of method 200 (*e.g.*, manually (*i.e.*, through interaction with an operator), automatically, or partially-automatically) to define a model 220 of at least a portion of at least one of the associated processes 110 (process step 225). Processor 205 is directed to store model 220 in memory 210, preferably representing at least a portion of process 110 mathematically. The mathematical representation defines one or more relationships among inputs and outputs of process 110.

According to an advantageous embodiment, model 220 is defined using the following discrete state space model form:

$$x_{k+1} = Ax_k + Bu_k \quad (1)$$

$$y_k = Cx_k + Du_k \quad (2)$$

wherein x_k, u_k, and y_k represent various states of modeled process 110, wherein k is a time period and $k+1$ is a next time period, and A, B, C, and D respectively represent measurable characteristics of modeled process 110 at any given time period.

Processor 205 is directed to define a data structure, such as lookup table 165, having a plurality of accessible fields (process step 230). An exemplary source code embodiment for performing this definition is attached as APPENDIX A, and incorporated herein by reference as if fully set forth herein, and that is written in Pascal. Depending upon the needs of the particular implementation, the contents of such accessible fields may suitably be nulled, defaulted, or otherwise initialized or used. Memory 210, directed by processor 205, maintains lookup table 165, preferably representing, at least in part, an AB0I matrix 235 and a feedback vector 240.

According to an advantageous embodiment, AB0I matrix 235 and feedback vector 240 have the following respective definitions:

$$\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} x_k \\ u_k \end{bmatrix} \quad (4)$$

wherein I and 0 respectively and illustratively represent an identity matrix and an null matrix, for the purpose of this illustrative model, to maintain, or hold, MV constant (illustrated with respect to FIGURE 3).

Processor 205 is directed to delineate mathematically a relationship among the above-given matrix 235 and vector 240 (process step 245), which according to an advantageous embodiment, has the following form:

$$z_{k+1} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} z_k \quad z_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix} \quad (5)$$

Processor 205 is directed to delineate mathematically a relationship among the above-given discrete state space model form and the Z vector 240 (process step 250), which, according to an advantageous embodiment, gives the following prediction form for any p interval, or point in the future:

$$\hat{Y}(k+p)|k = [CD] \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}^p z_k \quad (6)$$

Stated generally, use of Z vector 240 represents, or defines, mathematically, the relationship among the one or more inputs and outputs of modeled process 110.

For a variety of purposes, as above-stated, for monitoring and for control of process 110, it is desirable to decrease utilization of processing resources. This may be accomplished, in part, through a recognition that certain characteristics of process 110 are measurable (*e.g.*, appraising, assessing, gauging, valuating, estimating, comparing, computing, rating, grading, synchronizing, analyzing, etc.), whether or not such characteristics are dependent, independent, interdependent, or otherwise effected by other characteristics of the same process, a group of processes, a facility, a process

stage, a group of process stages, a sequence of processes or process stages, or the like. Many of these measurable characteristics have a range of possible values, which may or may not change, or vary, over time. It is desirable, in the present example, to determine an efficient prediction form ("EPF"), the range of values of which may suitably be maintained in lookup table 165.

Processor 205 is directed to populate ones of the accessible fields 255 of lookup table 165 with a range of possible values of at least one measurable characteristic associated with at least process 110 (process step 260). An exemplary source code embodiment for performing this population is attached as APPENDIX B, and incorporated herein by reference as if fully set forth herein, and that is written in Pascal. According to the illustrative embodiment, it is desirable to have future predictions available, or precalculated, which may suitably be stored as an array of points within lookup table 165. This collection of points may be referred to as PV-*blocking*, which may be given by the following form for any p_i interval, or point in the future:

$$\hat{Y}(k+pv-blocking)|k = \begin{bmatrix} \hat{Y}(k+p_1)|k \\ \hat{Y}(k+p_2)|k \\ \vdots \\ \hat{Y}(k+p_m)|k \end{bmatrix} \quad (7)$$

wherein i is the index for PV-*blocking*. The foregoing calculation may suitably be condensed into a product of EPF and Z_k, which may be given by:

$$\hat{Y}(k+pv-blocking)|k = [\mathrm{EPF}]Z_k \quad (8)$$

wherein EPF may be given by:

$$[EPF] = \begin{bmatrix} epf_1 \\ epf_2 \\ \vdots \\ epf_m \end{bmatrix} \quad (9)$$

wherein epf_i is independent from the feed back information contained in the Z vector and may therefore be calculated in advance and given by:

$$epf_i = [CD]\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}^{p_i} \quad \textbf{(10)}$$

In short, exemplary processor 205 uses model 220 iteratively, or incrementally, to populate lookup table 165 with k possible values, thereby defining a range of values. A v_k vector is formulated to conveniently calculate both Z_k and $y_{(k+pvblocking)|k}$ for different incremental k, which has the following form:

$$V_{k+1} = \begin{bmatrix} AB \\ 0I \\ EPF \end{bmatrix} Z_k \quad V_k = \begin{bmatrix} z_k \\ \hat{Y}(k+pvblocking)|k \end{bmatrix} \quad \textbf{(11)}$$

Turning momentarily to FIGURE 3, illustrated is an exemplary two-dimensional graphical representation of MV and PV curves in accordance with a use of lookup table 165 in accordance with the control system 100 of FIGURES 1a and 1b and the principles of the present invention. It should be noted, that FIGURES 1a, 1b, 2, and 3, along with the various embodiments used to describe the principles of the present invention in this patent document are illustrative only. To that end, alternate embodiments of model 220 may define any particular process, a group of processes, a facility, a process stage, a group of process stages, an interrelationship among, or a sequence of, processes or process stages, or some suitable portion or combination of any of the same. It should be further noted that a matrix structure was chosen for the EPF in this embodiment, however, alternate embodiments may use any appropriate data structure or dedicated circuitry to create a suitably arranged lookup array, or table, or the like. Such data structures and dedicated circuitry may be populated off-line, on-line or through some suitable combination of the same; likewise, such populated data structures and dedicated circuitry may be static, dynamic, modifiable, centralized, distributed, or any suitable combination of the same.

Those of ordinary skill in the art should recognize that the computer system 105 described using processor 205 and memory 210 may be any suitably arranged hand-held, laptop/notebook, mini, mainframe or super computer, as well as network combination of the same. In point of fact, alternate embodiments of computer system 205 may include, or be replaced by, or combined with, any suitable circuitry, including programmable logic devices, such as programmable array logic ("PALs") and programmable logic arrays ("PLAs"), digital signal processors ("DSPs"), field

programmable gate arrays ("FPGAs"), application specific integrated circuits ("ASICs"), very large scale integrated circuits ("VLSIs") or the like, to form the processing systems described and claimed herein. To that end, while the disclosed embodiments require processor 205 to access and to execute a stored task/instructions from memory to perform the various functions described hereabove, alternate embodiments may certainly be implemented entirely or partially in hardware. Conventional processing system architecture is more fully discussed in Computer Organization and Architecture, by William Stallings, MacMillan Publishing Co. (3rd ed. 1993); conventional processing system network design is more fully discussed in Data Network Design, by Darren L. Spohn, McGraw-Hill, Inc. (1993); and conventional data communications is more fully discussed in Data Communications Principles, by R. D. Gitlin, J. F. Hayes and S. B. Weinstein, Plenum Press (1992) and in The Irwin Handbook of Telecommunications, by James Harry Green, Irwin Professional Publishing (2nd ed. 1992). Each of the foregoing publications is incorporated herein by reference for all purposes.

APPENDIX A

```
{ System name: SPID                    }
{ BEGIN HDR                            }
{ FILE:  /lnk/spid/prv/src/sp_data.i.s }
{ VER/REV: 01,01                       }
{ DATE:   05/14/98                     }
{ Honeywell Trade Secret - Treat as Honeywell PROPRIETARY }
{ END HDR                              }

{(*:PCHP:}
unit SP_DATA;

interface
uses mc_data, urv_data;
{:PCHP:*)}

CONST
        SPID_VERSION  = 100.0000;       {current version}
```

```
max_n_blk           = 10;
max_n_den           = 5;
max_num_pin         = 6;
max_delay_def       = 200;
URV_tol             = 0.001;
ON                  = 1;
OFF                 = 0;
WARM                = 11;
FORCE               = 1;
AUTO                = 0;

D_GRT_THAN_MAXD        = 201;
BAD_NUM_OR_DEN         = 202;
BAD_DEN_INTEGRATOR     = 203;
NEGATIVE_DEN_COEF      = 204;
ZERO_GAIN              = 205;
ALLOC_S_Z_ERR          = 206;
BAD_MAXD_INT           = 207;
BAD_T_CON              = 208;

TYPE
    sp_block_arr     = array [1..max_n_blk] of single;
    sp_block_arr_ptr = ^sp_block_arr;
    coef_arr         = array [1..max_n_den] of single; { Predictor model F-polynom. coefficients }
    coef_arr_ptr     = ^coef_arr;
    coef_arrs        = array [1..max_n_den * max_num_pin] of single; { Predictor model F-polynom. coefficients }
    sp_pin_arr_i     = array [1..max_num_pin] of integer;
    sp_pin_arr_s     = array [1..max_num_pin] of single;

    sp_pool_data_ptr = ^sp_pool_data;
    sp_cds_data_ptr = ^sp_cds_data;
    sp_cds_data = record
```

```
{--------------------------------------------------------------------
-- Section 1: Model Data Input Seciton (must have init value/BLD_visible)
-------------------------------------------------------------------- }

        {continuous model}
        {spid_cds:CAL}
        n_dv          : single;                { No. of DVs }
{used as integer}
        {spid_cds:CALC1(1..6)}
        n_num         : sp_pin_arr_s; { No. of B-poly coefficients }
        {used as integer}
        {spid_cds:CALC2(1..6)}
        n_den         : sp_pin_arr_s; { No. of F-poly coefficients }
        {used as integer}
        {spid_cds:CONV1(1..30)}
        num                : coef_arrs;        { Predictor model
B-polynom. coefficients }
        {spid_cds:CONV2(1..30)}
        den                : coef_arrs;        { Predictor model
F-polynom. coefficients }
        {spid_cds:CD(1:.6)}
        delay              : sp_pin_arr_s; { Dead-time (continuous) }
                {used as integer}
        {spid_cds:CC(1..6)}
        G_s                : sp_pin_arr_s; { Steady-state gains, used by
controller }
        {spid_cds:CMD(1..6)}
        max_delay          : sp_pin_arr_s; { Max dead-time (continuous) }
                {used as integer}
        {spid_cds:CPROF}
        perf_rat    : single;     { Performance ratio - feedback }
        {spid_cds:CTREND3}
        Three_Tau          : single;                { Three Tau
```

```
(continuous) }                                  {used as integer}

        {discrete model}
        {spid_cds:DEF1(1..6)}
        n_b                : sp_pin_arr_s; { # of num coef (discrete) }   {used as integer}
        {spid_cds:DEF2(1..6)}
        n_f                : sp_pin_arr_s; { # of dnum coef (discrete) }  {used as integer}
        {spid_cds:DEMNDIN1(1..30)}
        b_z                : coef_arrs;    { num coefs (discrete) }
        {spid_cds:DEMNDIN2(1..30)}
        f_z                : coef_arrs;    { dnum coefs (discrete) }
        {spid_cds:DDD(1..6)}
        d                  : sp_pin_arr_s; { Dead-time (intervals) }
                {used as integer}
        {spid_cds:DDFPNT(1..6)}
        max_d_int          : sp_pin_arr_s; { maxd / T_con; (intervals) }  {used as integer}
        {spid_cds:TSLOAD}
        T_con              : single;       { Control execution interval (minutes) }
        {spid_cds:NUMSTIL}

{------------------------------------------------------------------
-- Section 2: Variables used by Pascal/CL/LCN GUI
------------------------------------------------------------------}

        {spid_cds:YY_L}
        y_L_ent            : single;       {Entered CV low limit.}
```

```
{spid_cds:YY_H}
y_H_ent             : single;          {Entered CV high
limit.}
{spid_cds:Y_L}
y_L                 : single;          { CV low limit
}
{spid_cds:Y_H}
y_H                 : single;          { CV high limit
}
{spid_cds:YY}
y                   : single;          { Current source
value      }
{spid_cds:Y_NOTOK}
y_not_ok    : single;          { 1: -> y value is uncertain/out of
range  }
{spid_cds:YVAL1}
first_on            : single;          { first time after pv
service }
{spid_cds:YSEL}
y_hat       : single;
{spid_cds:Y1}
y_last      : single;          { Last good y value        }
{spid_cds:YVAL2}
y_Filt_Bias         : single;          { Filtered y-bias for bias
updating,Pool? }
{spid_cds:YVAL3}
Filt_Const     : single;       {Bias correction Filter Factore,
Pool?  }
{spid_cds:YLDNEW}
y_Filt_Ramp  : single;
{spid_cds:YLDTGT}
y_Filt_Ramp2  : single;
{spid_cds:YIELD}
Filt_opt_Const : single ;
```

```
{spid_cds:YIELDINT}
Filt_opt_Const2: single ;
{spid_cds:Y2}
y_last_BAD          : single;          { 1=> Last y is BAD, 0=> Good ,P}
{spid_cds:TYPE}
y_type        : single;          { 0: stable, 1:integrator, LCN GUI uses it }
{spid_cds:Y_PRI}
y_UFP0      : single;
{spid_cds:Y_NEWVAL(1..21)}
y_UFP            : array [0..20] of single;      {       Unforced prediction}
{spid_cds:UNDIV01}
u_L_ent            : single;          {Entered MV low limit.}
{spid_cds:UNDIV02}
u_H_ent            : single;          {Entered MV high limit.}
{spid_cds:U_L}
u_L                : single;          { MV low limit }
{spid_cds:U_H}
u_H                : single;          { MV high limit }
{spid_cds:LOWLIM}
du_L          : single;          { MV move low limit }
{spid_cds:HIGHLIM}
du_H          : single;          { MV move high limit }
{spid_cds:UCL(1..6)}
u                  :  sp_pin_arr_s;  { Current actual value }
```

```
{spid_cds:U1(1..6)}
um1                : sp_pin_arr_s;  { MV at previous interval }
{spid_cds:U_0}
u_0        : single;        { Output from controller }
{spid_cds:DIS_NAME}
ss_display    : single;     { PC use only. use in simulation. Flag: 0 means display ss values }
{spid_cds:STP}
saved_3t_pr   : single;     { saved Three_Tau ( for integrator ) or Perf_rat ( for stable ) }
{spid_cds:FLAGS}
setup_flag    : single;     { if = force, call setup; if = auto, don't call setup except necessory }

n_sJL         : single;     { No. of intervals for J.L.'s mods} {used as integer}
{spid_cds:BLOLVL(1..10)}
y_blk                : sp_block_arr;  { CV blocking values }
{spid_cds:BLOMNMIN}
y_num_blk            : single;   { total # of blocking intervals }
{spid_cds:BLOTIPC}
y_num_start      : single;     { starting # of actual blocking intervals }
{spid_cds:BLOMIN}
y_num_end            : single;   { actual # of blocking intervals }
{spid_cds:DIFPRXX(1..5)}
dx                   : coef_arr;   { used in spidScalc}
{spid_cds:ZOOMDEV(1..5)}
z                          : coef_arr;   { used in spidScalc}
```

```
{spid_cds:NUMSAMP}
n_mvblk          : single;          { mv block sampling time } {used as integer}
{spid_cds:BLRQ(1..10)}
mv_blk                    : sp_block_arr; { used in spidScalc}
{rmpc_cds:MODFACT}
controller_mode      : single;      {only on DLL for now, 0-off;1-on;11-warm}
{spid_cds:VVP}
version                : single;      {save the release version in cds}
{spid_cds:INITPASS}
initpass             : single;        { Initialization pass counter }
{spid_cds:COUNTER}
predi_counter : single;            {useful in debugging, the same as kt}
{spid_cds:D1}
now_alarm            : single;          { Current alarm indicators }
{spid_cds:QANAPNT(1..10)}
QMapcds                     : sp_block_arr;
{spid_cds:ENTMPO11(1..150)}
epf1cds                     : array [1..max_n_den * max_n_den * max_num_pin] of single;
{spid_cds:ENTMPO12(1..300)}
epf2cds      : array [1..max_n_blk * max_n_den * max_num_pin] of single;
{spid_cds:DY(1..51)}
dys                         : array [0..max_delay_def] of single; { Output from controller }
{spid_cds:CDPNT}
myloc                 : sp_cds_data_ptr;{ Initialization pass counter}
```

```
        {spid_cds:SPARE(1..30)}
        spares                  : array [1..30] of single;

{----------------------------------------------------------------------
-- Section 3: CL/LCN GUI Only
----------------------------------------------------------------------}

        {spid_cds:TIMERQST}
        time_diff               : single;              { The amount of
CPU time in milliseconds   }

    end;

    {KYS - save the whole QMap or QMap1 which is the 1st row of QMap ? }
    sp_pool_data = record
        y_end                   : sp_block_arr_ptr;
        n_s                                 : integer;
        s_z                                 : heap_array_s_ptr;
        epf1                          : Matrix_Type;
        epf2                          : Matrix_Type;
        QMap                          : Matrix_Type;
    end;

{(*:PCHP:}
implementation
end.
{:PCHP:*)}

{ end sp_data }
```

APPENDIX B

```
{ System name: SPID                    }
{ BEGIN HDR                            }
{ LCN FILE: /lnk/rmpc/prv/src/sp_proc.s            }
{ PC FILE: sp_proc.pas                 }
{ DATE: 05/20/98                       }
{ Honeywell Trade Secret - Treat as Honeywell PROPRIETARY }
{ END HDR                              }

{(*:PCHP:}

unit SP_PROC;

{$N+,E-    8087 IEEE floating-point, no emulation}
{$A+,B-,G+   Word align, Boolean shortcut, protected mode}
interface
uses sp_data, mc_data {:PC_USE:}, urv_data {:}, mc_lib, mc_rca, mem_mnag;

procedure sp_ctfcheck(model_no       : integer;
                      n_dv           : integer;
                      var n_num      : single;
                      var n_den      : single;
                      var num        : coef_arrs;
                      var den        : coef_arrs;
                      var G_s        : single;
                      delay          : single;
                      var max_delay  : sp_pin_arr_s;
                      var max_d_int  : sp_pin_arr_s;
                      T_con          : single;
                      var setup_flag : single;
                      Three_Tau      : single;
                      perf_rat       : single;
                      y_type         : single;
                      var saved_3t_pr : single;
```

```
                        var call_setup        : single;
                        var status            : single);

procedure get_3tau (n_den                    : integer;
                        var den               : coef_arrs;
                        var Three_Tau         : single;
                        var status            : single);

procedure sp_blocker ({ Inputs -- }
                            delay_int    : single;
                        { Dead-times. }
                            n_sJL        : single;
                        { Number of intervals for J.L.'s mods.}
                            perf_rat               : single;
                        { FF cntl/FB cntl response ratio.}
                        { Outputs -- }
                            var y_blk      : sp_block_arr;
                        { CV block intervals (1 is current).}
                            var y_num_blk  : single;
                        { No. of block intervals per CV.}
                            var Rtc          : single;
                        { the beginning of control horizon}
                            var status: single
                        { Status of blocker calcs});

Procedure spid_calc_S_z (n_B                 : single;
                            var B_z              : coef_arrs;
                            n_F                  : single;
                            var F_z          : coef_arrs;
                            kp                   : integer;
                            var S_z              : heap_array_s_ptr;
                            var status       : single);

procedure sp_ctrans     (model_no         : integer;
```

```
        n_num               : single;
        n_den               : single;
        var num             : coef_arrs;
        var den             : coef_arrs;
        G_s                 : single;
        T_con               : single;
        {output}
        var n_b             : single;
        var n_f             : single;
        var b_z             : coef_arrs;
        var f_z             : coef_arrs;
        var status          : single);

procedure sp_dtf2epfp   (n_b        : single;
        n_f                 : single;
        var b_z             : coef_arrs;
        var f_z             : coef_arrs;
        y_num_end           : integer;
        var y_end           : sp_block_arr;
        delay_int           : single;
        lcnunit             : integer;
        n_order             : integer;
        {output}
        var epf1            : Matrix_Type;
        var epf2            : Matrix_Type;
        var status          : single);

procedure new_alloc_urv (var urv    : urv_set;
    tot_num_cv, {will include num of combined constraints for
    DQP}
        num_mv              : integer;
        callerID            : integer; {see URV_DATA.PAS
            for definition}
        lcnunit {:LCN_USE: : $unit_identifier; :}
```

```
                    {:HP_USE:    : integer;          :}
                    {:VAX_USE:   : integer;       :}
                    {:GHS_USE:   : integer;       :}
                {:PC_USE:}   : integer;  {:}{Point's unit }     var  status  :
                      single);
                          (*:LOCAL:*)

procedure new_release_urv(var urv : urv_set;
                                        lcnunit     {:LCN_USE:          :
$unit_identifier;  :}

                        {:HP_USE:    : integer;             :}

                        {:VAX_USE:   : integer;       :}

                        {:GHS_USE:   : integer;       :}

                        {:PC_USE:}   : integer;      {:}{ Point's unit }
                                            var     status     :     single);
(*:LOCAL:*)

procedure sp_h2solut (n_s                                  : integer;
                  var n_mvblk : single;
                       var mv_blk     :       sp_block_arr;
                       var s_z             : heap_array_s_ptr;
                       y_num_end           : integer;
                       var y_end           :  sp_block_arr;     {   CV   block
intervals (1 is current).   }
                       delay_int           : single;
                       y_type              : single;
                       G_s                                 : single;
                       n_sJL                          : single;
                       lcnunit             : integer;
                       override_tol: single;
```

```
var QMap : Matrix_Type;
var status : single
);

implementation
{:PCHP:*)}

{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure sp_ctfcheck( model_no : integer;
                       n_dv : integer;
                       var n_num : single;
                       var n_den : single;
                       var num : coef_arrs;
                       var den : coef_arrs;
                       var G_s : single;
                       delay : single;
                       var max_delay : sp_pin_arr_s;
                       var max_d_int : sp_pin_arr_s;
                       T_con : single;
                       var setup_flag : single;
                       Three_Tau : single;
                       perf_rat : single;
                       y_type : single;
                       var saved_3t_pr : single;
                       var call_setup : single;
                       var status : single);
var
        i, j, total_max_d_int, first_non_zero_ind: integer;
begin

        if status = 0 then
```

```
begin {status = 0}

    { if (call_setup = force) or the condition below is satisfied, call setup }

    if ( call_setup = AUTO ) and
        ( ( setup_flag = FORCE ) or
            ( (y_type = 0) and (perf_rat <> saved_3t_pr) ) or
            ( (y_type = 1) and (Three_Tau <> saved_3t_pr) ) ) then
        call_setup := FORCE;

    if call_setup=FORCE then
        begin {call_setup}

            while (n_den > 0 ) and ( den[round(n_den)] = 0 ) do
                n_den := n_den - 1;

            while (n_num > 0 ) and ( num[round(n_num)] = 0 ) do
                n_num := n_num - 1;

            if (delay > max_delay[model_no]) then
                status := D_GRT_THAN_MAXD

            else if T_con <= 0 then
                status := BAD_T_CON

            else if (n_num > max_n_den) or (n_den > max_n_den) or (n_num >= n_den)
                or (n_num < 1) or (n_den < 2) then
```

```
status := BAD_NUM_OR_DEN;

if status = 0 then
    begin {n_num and n_den are inside of ranges}

        first_non_zero_ind := 1;
        while ( first_non_zero_ind <= n_den ) and
        ( den[first_non_zero_ind] = 0 ) do
            first_non_zero_ind := first_non_zero_ind + 1;

        max_d_int[model_no] := round(max_delay[model_no] / T_con);

        total_max_d_int := 0;
        for j := 1 to (n_dv+1) do
            total_max_d_int := total_max_d_int +
                round(max_d_i
                nt[j]);

        if ( total_max_d_int > max_delay_def ) then
            status := BAD_MAXD_INT

        else if ( first_non_zero_ind > n_den ) or
        ( first_non_zero_ind > 3 ) then
            status := BAD_DEN_INTEGRATOR
```

```
else
    begin
        for j := (first_non_zero_ind+1) to round(n_den) do
            den[j] := den[j] / den[first_non_zero_ind];
        G_s := G_s * den[first_non_zero_ind];
        den[first_non_zero_ind] := 1;
        for i := 2 to round(n_den) do
            if den[i] < 0 then
                status := NEGATIVE_DEN_COEF;
    end;

if status = 0 then
    if num[1] = 0 then
        status := ZERO_GAIN
    else
        begin
            for j := 2 to round(n_num) do
                num[j] := num[j] / num[1];
            G_s := G_s / num[1];
            num[1] := 1;
        end;
```

```
                    end;  {n_num and n_den are inside of
ranges}

               end;          {call_setup}

          end;  {status = 0}

end;

{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure get_3tau  (

     n_den                              : integer;
                                              var
den                                   : coef_arrs;
                                              var
Three_Tau               : single;
                                              var
status                  : single);

var
     i            : integer;
     mt           : single;
begin

     if status = 0.0 then
          begin  {status = 0.0}

               Three_Tau := 0;
               for i := 1 to n_den-1 do
                    begin  {for}
                         if i = 1 then
```

```
                mt := den[2]
            else if i = 2 then
                mt := sqrt(den[3])
            else if i = 3 then
                mt := exp( ln( den[ 4 ] ) / 3 )
            else
                mt := sqrt( sqrt( den[5] ) );

            if mt > Three_Tau then
                Three_Tau := mt;

        end;    {for}

        Three_Tau := Three_Tau * 3;

    end;    {status = 0.0}
end;


{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
Procedure spid_calc_S_z( n_B : single;
                                    var B_z : coef_arrs;
                                    n_F : single;
                                    var F_z : coef_arrs;
                                    kp  : integer;
                                    var S_z        : heap_array_s_ptr;
                                    var status : single
                                    ); (*:LOCAL:*)

Var
    loc_y               : array[-n_F_max_dim..0] of single;
    total_B, term1, term2    : single;
```

```
        i, i_kp             : integer;

    Begin {spid_calc_S_z}
        if status = 0.0 then
            begin   {status = 0}
                {Calc total_B to save calculations later.}
                total_B := 0.0;
                for i := 1 to round(n_B) do
                    total_B := total_B + B_z[i];

                {Initialize loc_y array to 0}
                for i := 0 to round(n_F) do
                    loc_y[-i] := 0.0;

                for i_kp := 1 to kp do
                    begin {i_kp}

                        {First, shift ("age") loc_y array}
                        for i := - round(n_F) to -1 do
                            loc_y[i] := loc_y[i+1];

                        {Then, find term1 - the contribution of B_z}
                        if i_kp > round(n_B) then
                            term1 := total_B
                        else
                            begin
                                term1 := B_z[1];
                                for i := 2 to i_kp do
                                    term1 := term1 + B_z[i];
                            end;

                        {Next, calculate term2 - the contribution of F_z}
                        term2 := 0.0;
                        for i := 1 to round(n_F) do
```

```
                    term2 := term2 - F_z[i] * loc_y[-i];

                {Finally, put current y into loc_y[0]}
                loc_y[0] := term1 + term2;
                s_z^[i_kp] := loc_y[0];

            end;  {i_kp}
        end; {status = 0}
End;  {spid_calc_S_z}


{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}

procedure sp_blocker (
    { Inputs -- }
    delay_int           : single  ;              {              Dead-times.
}
    n_sJL           : single  ;              {  Number  of  intervals  for  J.L.'s
mods. }
    perf_rat            : single ;               { FF  cntl/FB  cntl  response
ratio.  }
    { Outputs -- }
    var y_blk       : sp_block_arr   ;  { CV block intervals (1 is current).   }
    var y_num_blk   : single                    ;  { No. of block intervals per CV .
}
    var Rtc: single       ;  { the beginning of control horizon     }
    var status: single     { Status of blocker calcs           }
    );

var
    Rte, Rti            : single;
    i, nkvi                                     : integer;
    Iv_temp                                     : cv_block_arr;
```

```
        Kv_temp                                          : cv_block_arr;

    begin  { Start blocking calculations }
        if status = 0.0 then
            begin  {status = 0}
                {for SISO, Rti = Resp_time = n_sJL }
                Rti := n_sJL + delay_int;
                Rtc := minr(1, perf_rat) * n_sJL + delay_int;
                Rte := ( Rti + Rtc ) / 2;

                GetPVblk( Kv_temp,
                                        Iv_temp,
                                        nkvi,
                                        round(delay_int),
                                        Rtc, Rte,
                                        10,
                                        1,
                                        1,
                                        false,
                                        1.0);

                for i := 1 to nkvi do
                    y_blk[i] := Kv_temp[i];

                y_num_blk := nkvi;
            end;   {status = 0}
    end;  { End blocking calculations }


{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure sp_ctrans    (model_no                           : integer;
                        n_num                              :
single;
```

```
                n_den                                   :
single;
                var num                                 :
coef_arrs;
                var den                                 :
coef_arrs;
                G_s                          : single;
                T_con                        : single;
                {output}
               var n_b                                  :
single;
               var n_f                              : single;
               var b_z                              : coef_arrs;
               var f_z                              : coef_arrs;
               var status                   : single);

var
     n_numi, n_deni                                 : integer;
     sqtau2, tau1mul4, tmid1, tmid2, tmid3, tmid4, tmid5, tmid6, tmid7, tmid8, a, b
     : single;
     numptr, denptr, b_z_ptr, f_z_ptr : coef_arr_ptr;

begin  { ctrans }
     if status = 0.0 then
          begin  {status = 0}
               n_numi          := round(n_num);
               n_deni := round(n_den);

               if (n_numi = 1) and (n_deni = 2) and (den[1] = 0) then
                    begin { 1/S }
                         n_b := 1;
                         n_f := 1;
                         b_z[1] := G_s * T_con;
                         f_z[1] := -1;
```

```
end   { 1/S }

else if (n_numi = 1) and (n_deni = 2) and (den[1] = 1 ) then
    begin           { 1/( t1 * S + 1 ) }
        n_b := 1;
        n_f := 1;
        b_z[1] := G_s * (1 - exp( -T_con / den[2]));
        f_z[1] := - exp(-T_con / den[2]);
    end             { 1/( t1 * S + 1 ) }

else if (n_numi = 1) and (n_deni = 3) and (den[1] = 1) then
    begin           { 1/( t1 * S^2 + t2 * S + 1) }
        n_b := 2;
        n_f := 2;
        sqtau2 := sqr ( den[2] );
        tau1mul4 := den[3] * 4;
        tmid1    := sqrt(abs( sqtau2 - tau1mul4 ));

        if ( tmid1 < eps ) then
            begin
                tmid2 := T_con / sqrt(den[3]);
                tmid3 := exp (-tmid2);
                b_z[1] := G_s * ( 1 - tmid3 * (1 +
tmid2) );
                b_z[2] := G_s * tmid3 * (tmid3 +
tmid2 - 1);
                f_z[1] := (-2) * tmid3;
                f_z[2] := tmid3 * tmid3;
            end
        {kys - think about sqtau2 = tau1mul4}
        else if ( sqtau2 < tau1mul4 ) then
            begin  { Tau2^2 less then ( 4 * Tau1 ), w0
}
                tmid8 := den[3] * 2;
```

```
            tmid2 := tmid1 / tmid8;
            tmid7 := tmid2 * T_con;
            tmid5 := cos(tmid7);
            tmid6 := sin(tmid7);
            tmid3 := den[2] / tmid1 * tmid6;
            tmid4 := exp ( - den[2] / tmid8 * T_con);
            b_z[1] := G_s * ( 1 - tmid4 * ( tmid5 + tmid3 ) );
            b_z[2] := G_s * ( sqr(tmid4) + tmid4 * ( tmid3 - tmid5) );
            f_z[1] := -2 * tmid4 * tmid5;
            f_z[2] := sqr( tmid4 );
        end          { Tau2^2 less then ( 4 * Tau1 ), w0 }
    else if ( sqtau2 > tau1mul4 ) then
        begin        { Tau2^2 greater then ( 4 * Tau1 ), ab }
            a := (den[2] - tmid1) * 0.5 / den[3];
            b := (den[2] + tmid1) * 0.5 / den[3];
            tmid2 := tmid1 / den[3];     {b-a}
            tmid3 := exp ( - a * T_con ) ;
            tmid4 := exp ( - b * T_con ) ;
            b_z[1] := G_s * (b * (1-tmid3) - a * (1-tmid4)) / tmid2;
            b_z[2] := G_s * (a * (1-tmid4) * tmid3 - b * (1-tmid3) * tmid4) / tmid2;
            f_z[1] := - tmid3 - tmid4;
            f_z[2] := exp ( - den[2] / den[3] * T_con );
        end;         { Tau2^2 greater then ( 4 * Tau1 ), ab }
```

```
                    end          { 1/( t1 * S^2 + t2 * S + 1) }

                else
                    begin
                        {(*:PC_ONLY:}
                        writeln('Ctrans cannot do it now');
                        {:PC_ONLY:*)}
                        {status := 100; }
                    end;

            end;  {status = 0}

end;  { ctrans }


{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}

{[epf1, epf2]=dtf2epfp(mn,md,pv_end-mdt,0) }
procedure sp_dtf2epfp   (
                                                            n_b
                                                : single;
                                                            n_f
                                                : single;
                                                            var
b_z                             : coef_arrs;
                                                            var
f_z                             : coef_arrs;

        y_num_end               : integer;
                                                            var
y_end               : sp_block_arr;

        delay_int                               : single;
```

```
        lcnunit                                    : integer;
        n_order                                    : integer;
        {output}
                                                                    var
    epf1                                  : Matrix_Type;
                                                                    var
    epf2                                  : Matrix_Type;
                                                                    var
    status                           : single);

    var
        pvrow                                  : Row;
        n1, get_amount, i, j, i1, j1        : integer;
        code : single;
        gen_ptr       : r_anyptr;

    begin { sp_dtf2epfp }
        if status = 0.0 then
            begin {status = 0}
                code := 0.0;

                for i := 1 to n_order do
                    begin
                        epf1.mat[i]^[1] := -f_z[i];
                        for j := 2 to n_order do
                            if (j=i+1) then
                                epf1.mat[i]^[j] := 1
                            else
                                epf1.mat[i]^[j] := 0;
                    end;
```

```
if status = 0.0 then
    begin  {no error after sp_tf2obsv}

        for i := 1 to n_order do
            epf1.mat[i]^[epf1.m] := b_z[i];

        for j := 1 to n_order do
            epf1.mat[epf1.n]^[j] := 0.0;

        epf1.mat[epf1.n]^[epf1.m] := 1;

        for j := 1 to epf1.n do
            pvrow[j] := epf1.mat[1]^[j];

        i := 1;
        for j := 1 to round(y_end[y_num_end]-delay_int)
do
            begin
                PreVectMult (pvrow, epf1,
epf2.mat[i]^);
                for j1 := 1 to epf1.m do
                    pvrow[j1] :=
epf2.mat[i]^[j1];
                if ( j = round(y_end[i]-delay_int) )
then
                    i := i + 1;
            end;

        epf1.n := epf1.n -1; {taking out the last row, but
leaving the memory}

    end;  {no error after sp_tf2obsv}
  end; {status = 0}
end; { sp_dtf2epfpcal }
```

```
{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}

{--------------------------------------------------------------------------------}
{Procedure to allocate space for U, R, and V.  Called from setup  }
{and from rca_body (in order to get second URV for PreRCA)        }
{--------------------------------------------------------------------------------}
procedure new_alloc_urv (    var urv    : urv_set;

        tot_num_cv, {will include num of combined constraints for DQP}

        num_mv     : integer;

        callerID   : integer; {see URV_DATA.PAS for definition}

        lcnunit   {:LCN_USE:     : $unit_identifier;  :}

                                   {:HP_USE:      : integer;            :}

                                   {:VAX_USE:     : integer;        :}

                                   {:GHS_USE:     : integer;        :}

                                   {:PC_USE:}     : integer;        {:}{ Point's unit }
                                                                         var
status : single);         (*:LOCAL:*)

var
        n,m,i   : integer;
        code    : single;

begin {new_alloc_urv}
```

```
code := 0.0;
with urv do
        begin  {with urv}
                {Based on callerID, calculate n and m.}
                if (callerID = CID_RMPCT) or (callerID = CID_RMPCT_PRERCA) then
                        begin
                                {later, might determine different (smaller) size for PreRCA}
                                n := maxi(tot_num_cv * max_cv_blk + num_mv * (max_mv_blk-1), tot_num_cv * 2 + num_mv + 1);
                                {for dynamic CV rows}   {for dynamic CX rows}   {for SS CV row + extra LP/QProws}
                                m := num_mv * max_mv_blk;
                        end
                else if (callerID = CID_DQP) then
                        begin
                                n := tot_num_cv * 2 + num_mv + 1;
                                {for SS CV row + extra LP/QProws}
                                m := num_mv;
                        end
                else if (callerID = CID_SPID) then
                        begin
                                n := tot_num_cv;
                                m       := num_mv;
                        end
                else
                        begin
                                n := 0;
                                m := 0;
                        end;

                n := mini(max_row_size, n);
```

```
{double check MC_Control/OptSS later, what if n > max_row_size???}

{store size allocated for later use in release_urv}
n_alloc := n;
m_alloc := m;

{grab space for U}
for i := 1 to n do
    U.mat[i] := (*:loophole:*) Row_ptr ( new_mc_getmem(n * sizeof(URV_sd),
                    lcnunit, code, status));

{grab space for R}
for i := 1 to n do
    R.mat[i] := (*:loophole:*) Row_ptr ( new_mc_getmem(m * sizeof(URV_sd),
                    lcnunit, code, status));

{grab space for V}
for i := 1 to m do
    V.mat[i] := (*:loophole:*) Row_ptr ( new_mc_getmem(m * sizeof(URV_sd),
                    lcnunit, code, status));
U.n:=0; U.m:=0;
R.n:=0; R.m:=0;
V.n:=0; V.m:=0;
R.mt := UpperTri;

  end; {with urv}
end; {new_alloc_urv}

{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure new_release_urv(var urv : urv_set;
                              lcnunit  {:LCN_USE:     : $unit_identifier;
:}

                          {:HP_USE:      : integer;              :}

                          {:VAX_USE:     : integer;         :}

                          {:GHS_USE:     : integer;         :}

                          {:PC_USE:}     : integer;         {:}{ Point's unit }
                              var status : single);   (*:LOCAL:*)
var
        n,m,i, get_amount : integer;
        gen_ptr         : r_anyptr;

begin {new_release_urv}
        with urv do
                begin  {with urv}
                        n := n_alloc;
                        m := m_alloc;
                        {grab space for U}
                        get_amount := n * sizeof(URV_sd);
                        for i := 1 to n do
                                begin
                                        gen_ptr := (*:loophole:*)r_anyptr(U.mat[i]);
                                        new_mc_freemem(gen_ptr,  get_amount,  lcnunit,
status);
                                end;

                        {grab space for R}
                        get_amount := m * sizeof(URV_sd);
                        for i := 1 to n do
```

```
                    begin
                        gen_ptr := (*:loophole:*)r_anyptr(R.mat[i]);
                        new_mc_freemem(gen_ptr, get_amount, lcnunit,
status);
                    end;

                {grab space for V}
                get_amount := m * sizeof(URV_sd);
                for i := 1 to m do
                    begin
                        gen_ptr := (*:loophole:*)r_anyptr(V.mat[i]);
                        new_mc_freemem(gen_ptr, get_amount, lcnunit,
status);
                    end;

        end; {with urv}
end; {new_release_urv}




{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure getmvblk (
                                                                var
nblk                    : integer;

        lastblk                 : integer;
                                                                dense
                                : integer;
                                                                var
mv_blk          : sp_block_arr
                                                                );
var
```

```
        i          : integer;
    begin  {getmvblk}

        if nblk >= max_n_blk then
            nblk := max_n_blk;

        if (lastblk <= nblk) then
            begin  { lastblk <= nblk }
                for i := 1 to lastblk do
                    mv_blk[i] := i;
                nblk := lastblk;
            end    { lastblk <= nblk }
        else
            begin  { lastblk > nblk }
                mv_blk[1] := 1;
                for i := 2 to nblk do
                    mv_blk[i] := 0;

                for i := 2 to nblk do
                    begin
                        mv_blk[i] := mv_blk[i] + ( exp (dense * i / nblk) -
1 ) *
                            ( lastblk / ( exp(dense) - 1 ) );
                        if mv_blk[i] > ( mv_blk[i-1] + 1 ) then
                            mv_blk[i] := round( mv_blk[i] )
                        else
                            begin
                                if i < nblk then
                                    mv_blk[i + 1] := mv_blk[i +
1] +
                                    ( mv_blk[i - 1] + 1 -
                                mv_blk[i] ) / 2;
                                mv_blk[i] := mv_blk[i - 1] + 1;
                            end;
```

```
            end;
      end;   { lastblk > nblk }

end;  {getmvblk}




{&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&}
procedure sp_h2solut (
                  n_s                          : integer;
                  var n_mvblk : single;
                  var mv_blk     :     sp_block_arr;
                  var s_z           : heap_array_s_ptr;
                  y_num_end         : integer;
                  var y_end         :  sp_block_arr;     {   CV   block
intervals (1 is current).  }
                  delay_int          : single;
                  y_type             : single;
                  G_s                              : single;
                  n_sJL                      : single;
                  lcnunit            : integer;
                  override_tol: single;
                  var QMap           : Matrix_Type;
                  var status  : single
                                    );

var
      return
                        : single;
```

```
i, j, dense, n_mvblki, last_blk            : integer;
urv
                                           : urv_set;
built_row
Row;                                                     :

{============================================================
=====================}
      {                    A                    }

{============================================================
=====================}
      {Finds a value in the A matrix, which is never explicitly built as a    }
      {complete matrix.  The only parameters to A are i and j, the row and    }
      {column indices in A.  A does, however, access several of the parameters
}
      {that are passed to the main RCA procedure that calls A.  These "global"
}
      {parameters are                                   }
      {                                              }
      {    pv_blk,                  --      single array, index is row or
column in our A,}
      {               value is index to entire matrix "S?" of step  }
      {               response data                          }
      {                                                    }
      {    delay_int      -- integer, value is dead time for each block    }
      {                                                    }
      {    S_z            -- array of step response polynomials}
      {                                                    }
      {    n_S            -- integer, value is number of values in
}
      {               step response polynomial              }
```

```
Function A(i,j : integer) : single;   (*:LOCAL:*)

var
        value : URV_sd;
        ind_S : integer;  {index within step response polynomial}

Begin {A}

        value := 0.0;

        {use the 5 tau value if process is unstable}
        {Q2J - verify}
        ind_S := round(y_end[i] - delay_int - j + 1);
        if ind_S > 0 then
                begin
                        if ind_S <= n_S then
                                value := S_z^[ind_S]
                        else if ( y_type = 1 ) then    {not stable}
                                value := S_z^[n_S] + G_s * (ind_S - n_S)
                        else
                                value := S_z^[n_S];
                end;

        {
        if ( (i-j+1) > 0 ) then
                value := S_z^[i-j+1];
        }
        A := value;

End; {A}
```

```
{---------------------------------------------------------------------}
{start of sp_h2solut procedure body}
{----------------------------------------------------------------------}

begin {sp_h2solut}
    if status = 0.0 then
        begin {status = 0}

            dense := 3;
            n_mvblki := round(n_mvblk);
            last_blk := round( minr(n_sJL, y_end[y_num_end] - delay_int) );
            getmvblk (n_mvblki, last_blk, dense, mv_blk);
            new_alloc_urv ( urv, y_num_end, n_mvblki, CID_SPID, lcnunit,
status);

            if status = 0.0 then
                with urv do
                    begin {with urv}
                        return := successful;
                        k := 0;

                        {QMap = pinv(Hss) = V * (R \ U') }
                        {URVAddRow returns U, R, and V, no
transpose }

                        for i := 1 to y_num_end do
                            begin
                                for j := 1 to n_mvblki do
                                    built_row[j] := A(i,
round(mv_blk[j]) );
                                {       built_row[j]      :=
A_blk(i,j);}
                                return                    :=
URVAddRow(U,R,V,k, built_row, n_mvblki);
```

```
end;

for i := 1 to R.n do
    begin
        for j := 1 to k do
            built_row[j] := U.mat[i]^[j];
            {R\U', pass U' collom which is U row into built_row,
            built_row is a used as collom}
        return := ForwardBackSub (R, built_row, k);
        { PostVectMult (V, built_row, QMap.mat[i]^);
        QMap is acturally QMap's transpose here
        We do not need to calculate the whole QMap on Am,
        but for offline, probably.
        }
        QMap.mat[1]^[i] := 0;
        for j := 1 to k do
        QMap.mat[1]^[i] := QMap.mat[1]^[i] +
            V.mat[1]^[j] * built_row[j];
        { QMap.mat[1]^[i] := QMap.mat[1]^[i] / ( n_s/n_mvblk );  }
    end;

    new_release_urv(urv, lcnunit, status);

  end; {with urv}
  n_mvblk := n_mvblki;
end;  {status = 0.0}
```

```
end;  {sp_h2solut}


{(*:PCHP:}
END.  {of unit}
{:PCHP:*)}
```

What is Claimed is:

1. A computer system for use with a process facility having a plurality of associated processes, comprising:

circuitry that is capable of maintaining a data structure having a plurality of accessible fields; and

a processor, associated with said circuitry, that is capable of populating ones of said plurality of accessible fields of said data structure with a range of possible values of at least one measurable characteristic associated with at least one process of said plurality of associated processes.

2. The computer system set forth in Claim 1 wherein said circuitry is capable of storing a task that directs said processor to populate said ones of said plurality of accessible fields of said data structure with said range of possible values.

3. The computer system set forth in Claim 1 wherein said circuitry is further capable of maintaining a model of at least a portion of said plurality of associated processes.

4. The computer system set forth in Claim 3 wherein said model includes a mathematical representation of at least a portion of said at least one process of said plurality of associated processes, said mathematical representation defining relationships among inputs and outputs of said at least one process of said associated processes.

5. The computer system set forth in Claim 3 wherein said processor is capable of using said model iteratively to populate ones of said plurality of accessible fields of said data structure with said range of possible values of said at least one measurable characteristic.

6. The computer system set forth in Claim 5 wherein said model includes at least one feedback variable representing, at least in part, an output of said at least one process of said associated processes.

7. The computer system set forth in Claim 6 wherein said processor

populates at least one of said plurality of accessible fields of said data structure in response to said at least one feedback variable of said at least one process of said associated processes.

8. The computer system set forth in Claim 3 wherein said model includes a manipulable variable.

9. The computer system set forth in Claim 8 wherein said processor is capable of at least substantially maintaining a value of said manipulable variable during at least a portion of said iterative population of said ones of said plurality of accessible fields of said data structure.

10. The computer system set forth in Claim 1, wherein said circuitry maintains statically said range of possible values of said at least one measurable characteristic associated with at least one process of said plurality of associated processes.

11. The computer system set forth in Claim 1 wherein said processor is further capable of using said range of possible values of said at least one measurable characteristic to predict an unforced response associated with said at least one process.

12. A method of operating a computer system that is for use with a process facility having a plurality of associated processes, said method of operation comprising the steps of:

maintaining a data structure having a plurality of accessible fields in circuitry associated with said computer system; and

populating ones of said plurality of accessible fields of said data structure using a processor, that is associated with said circuitry, with a range of possible values of at least one measurable characteristic associated with at least one process of said plurality of associated processes.

13. The method of operation set forth in Claim 12 further comprising the step of storing a task in said circuitry that is capable of directing said processor to populate said ones of said plurality of accessible fields of said data structure with said

range of possible values.

14. The method of operation set forth in Claim 12 further comprising the step of maintaining a model of at least a portion of said plurality of associated processes in said circuitry.

15. The method of operation set forth in Claim 14 wherein said model includes a mathematical representation of at least a portion of said at least one process of said plurality of associated processes, said mathematical representation defining relationships among inputs and outputs of said at least one process of said associated processes, and said method further comprises the step of using said model iteratively by said processor to populate ones of said plurality of accessible fields of said data structure with said range of possible values of said at least one measurable characteristic.

16. The method of operation set forth in Claim 15 wherein said model includes at least one feedback variable representing, at least in part, an output of said at least one process of said associated processes, and said method further comprises the step of using said processor, in response to said at least one feedback variable of said at least one process of said associated processes, to populate at least one of said plurality of accessible fields of said data structure.

17. The method of operation set forth in Claim 14 wherein said model includes a manipulable variable, and said method further comprises the step of at least substantially maintaining a value of said manipulable variable during at least a portion of said iterative population of said ones of said plurality of accessible fields of said data structure.

18. The method of operation set forth in Claim 12 wherein said circuitry maintains statically said range of possible values of said at least one measurable characteristic associated with at least one process of said plurality of associated processes.

19. The method of operation set forth in Claim 12 further comprising the

step of predicting an unforced response associated with said at least one process using said processor and said range of possible values of said at least one measurable characteristic.

20. A data structure for use with a computer system associated with a process facility having a plurality of associated processes, said data structure comprising a plurality of accessible fields, ones of said plurality of accessible fields maintaining a range of possible values of at least one measurable characteristic associated with at least one process of said plurality of associated processes.

21. The data structure set forth in Claim 20 wherein one of said plurality of accessible fields are capable of being selected by said computer system.

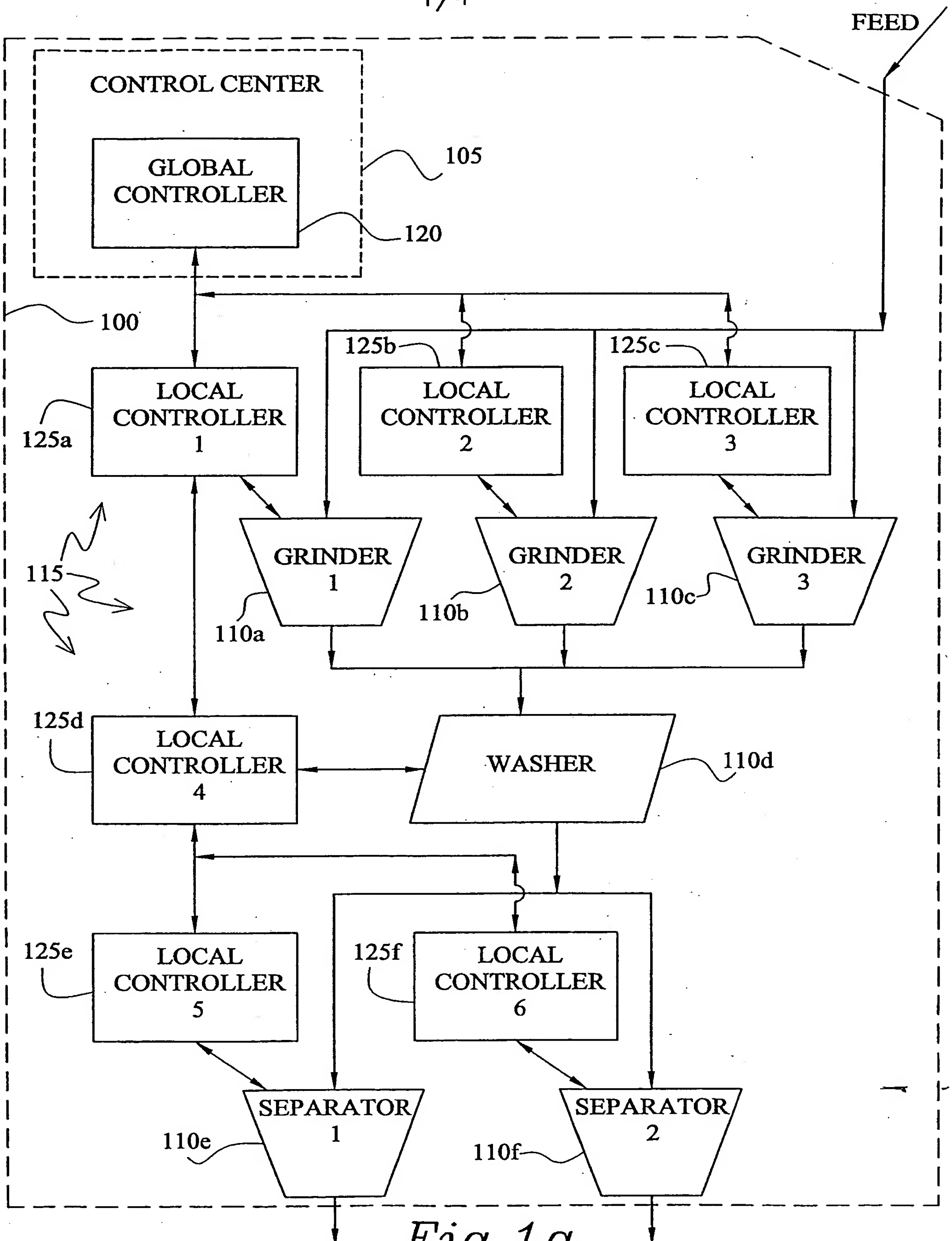

FEED
CONTROL CENTER
GLOBAL CONTROLLER
105
120
100
125b
125c
LOCAL CONTROLLER 1
LOCAL CONTROLLER 2
LOCAL CONTROLLER 3
125a
115
GRINDER 1
GRINDER 2
GRINDER 3
110a
110b
110c
125d
LOCAL CONTROLLER 4
WASHER
110d
125e
LOCAL CONTROLLER 5
125f
LOCAL CONTROLLER 6
SEPARATOR 1
SEPARATOR 2
110e
110f

Fig. 1a

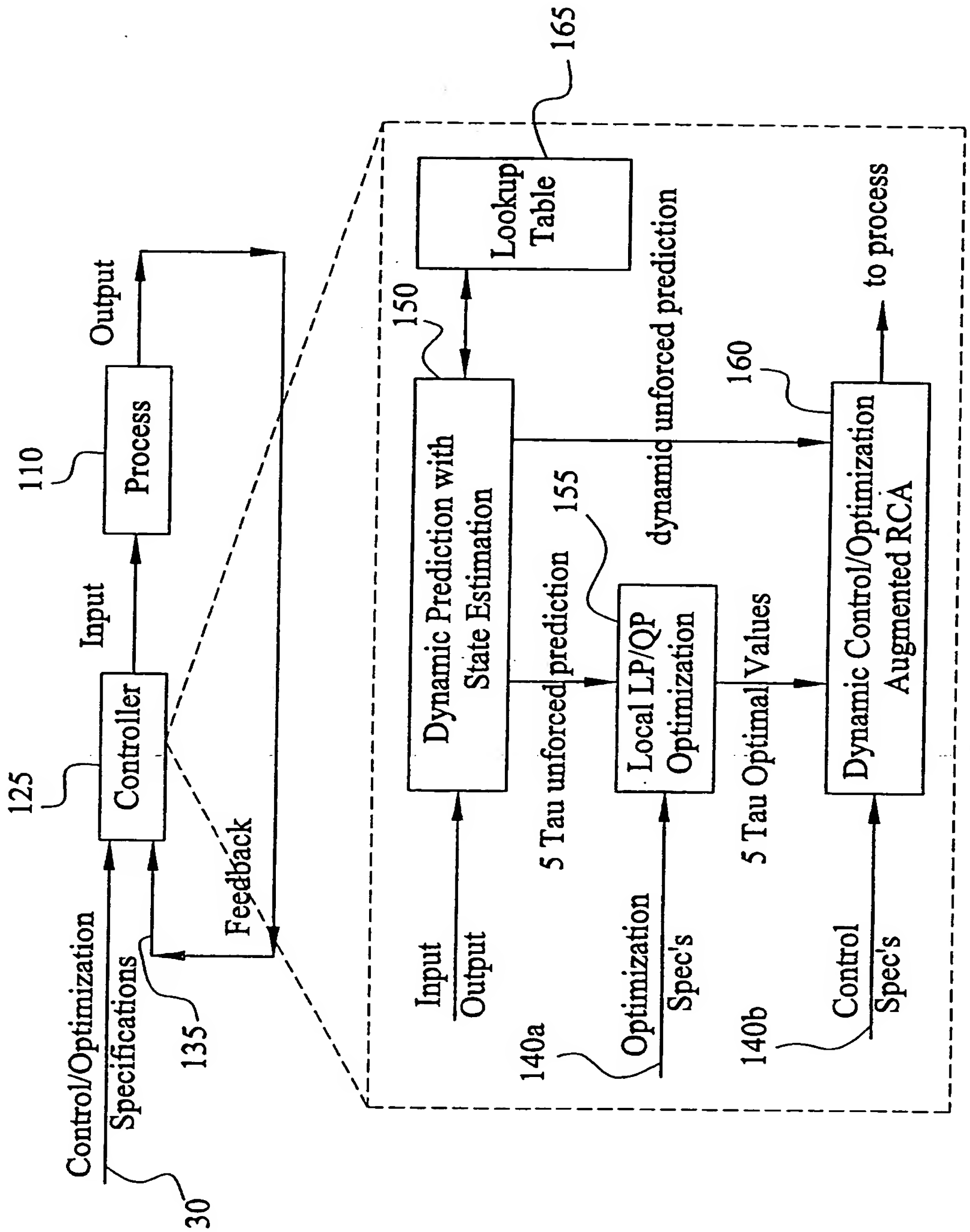

Control/Optimization Specifications
130
135
Feedback
125
Controller
Input
110
Process
Output
Input
Output
140a
Optimization Spec's
140b
Control Spec's
150
Dynamic Prediction with State Estimation
Lookup Table
165
5 Tau unforced prediction
155
Local LP/QP Optimization
5 Tau Optimal Values
dynamic unforced prediction
160
Dynamic Control/Optimization Augmented RCA
to process

Fig. 1b

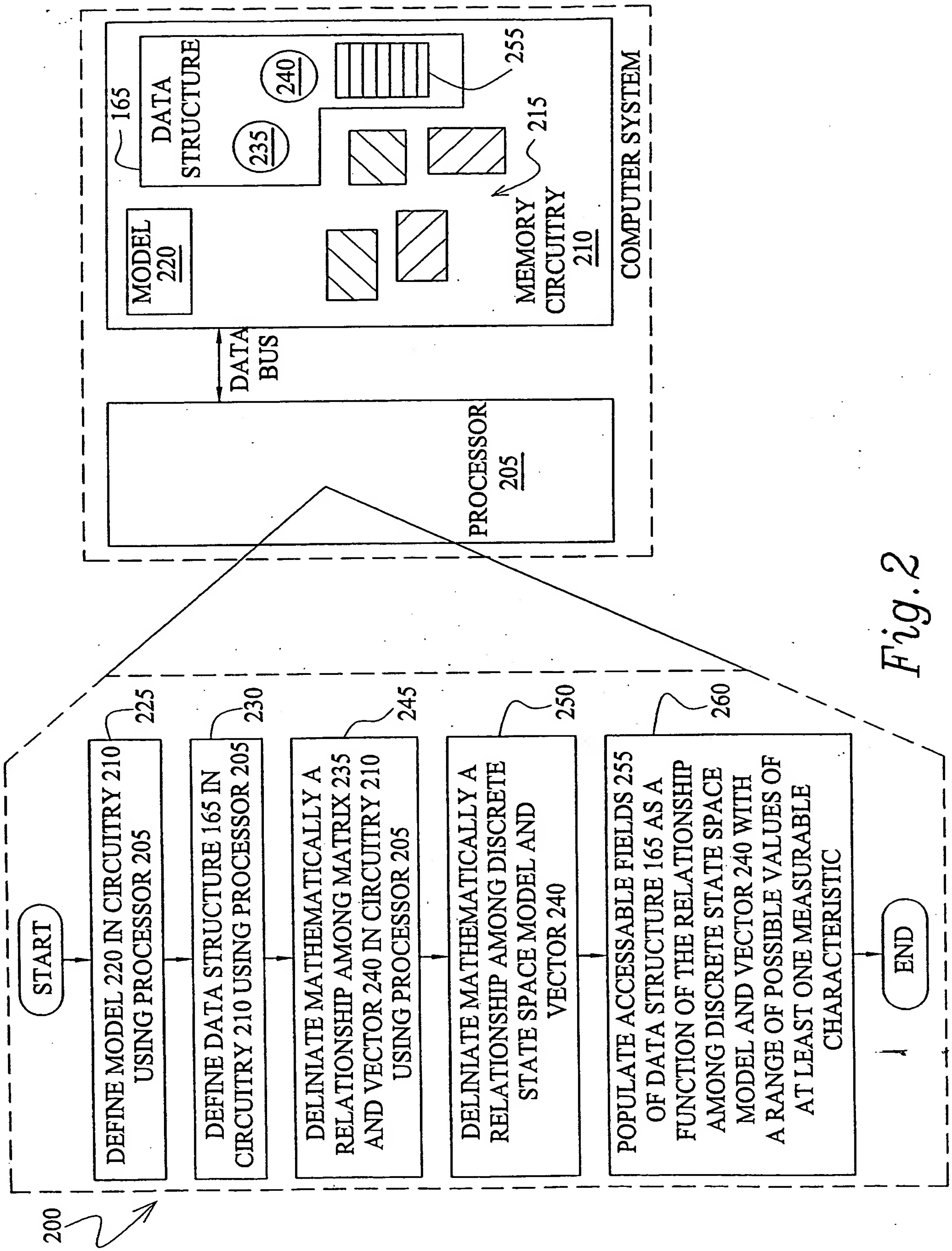

165
DATA STRUCTURE
235
240
255
MODEL 220
215
MEMORY CIRCUITRY 210
COMPUTER SYSTEM
DATA BUS
PROCESSOR 205
START
DEFINE MODEL 220 IN CIRCUITRY 210 USING PROCESSOR 205
225
DEFINE DATA STRUCTURE 165 IN CIRCUITRY 210 USING PROCESSOR 205
230
DELINIATE MATHEMATICALLY A RELATIONSHIP AMONG MATRIX 235 AND VECTOR 240 IN CIRCUITRY 210 USING PROCESSOR 205
245
DELINIATE MATHEMATICALLY A RELATIONSHIP AMONG DISCRETE STATE SPACE MODEL AND VECTOR 240
250
POPULATE ACCESSABLE FIELDS 255 OF DATA STRUCTURE 165 AS A FUNCTION OF THE RELATIONSHIP AMONG DISCRETE STATE SPACE MODEL AND VECTOR 240 WITH A RANGE OF POSSIBLE VALUES OF AT LEAST ONE MEASURABLE CHARACTERISTIC
260
END
200

Fig. 2

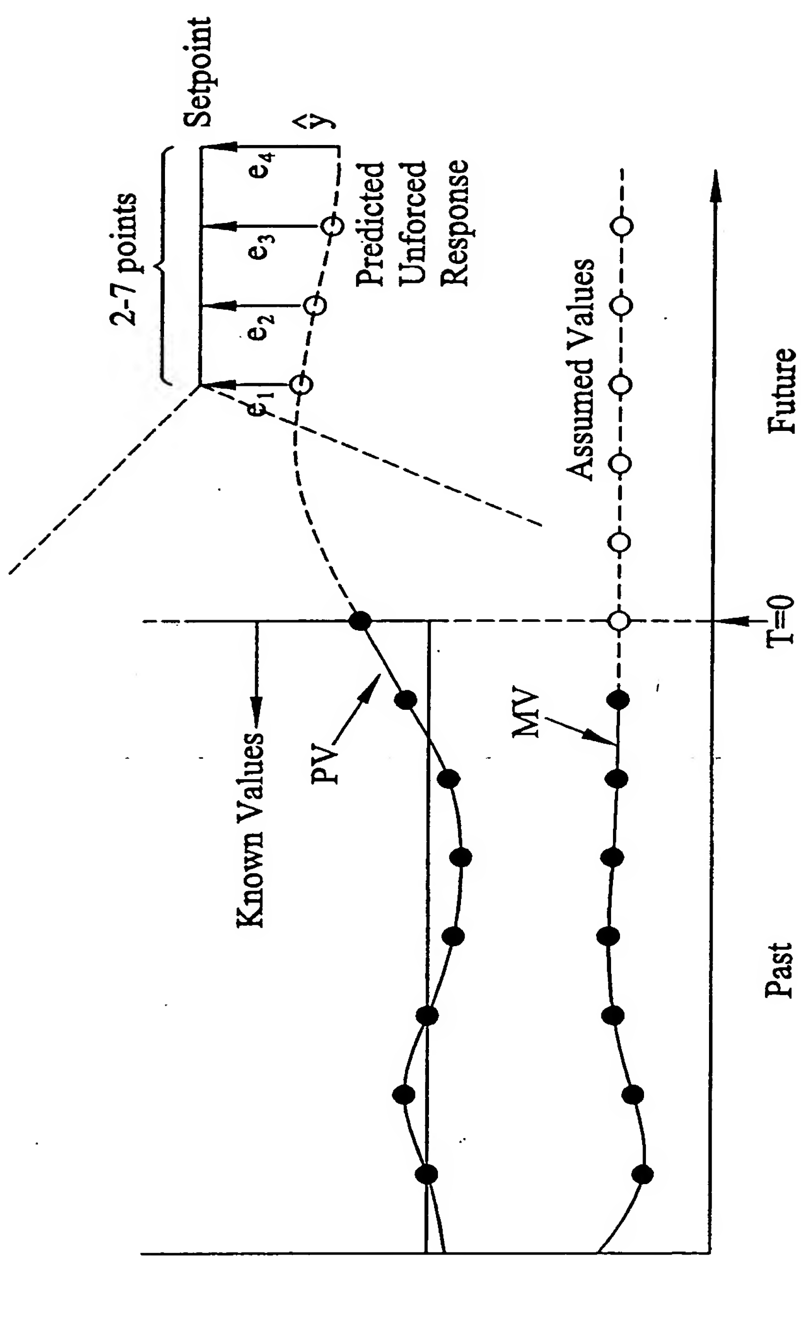

Setpoint
2-7 points
$\hat{y}$
e_1
e_2
e_3
e_4
Predicted
Unforced
Response
Assumed Values
Known Values
PV
MV
T=0
Past
Future

Fig. 3

INTERNATIONAL SEARCH REPORT

Int ...tional Application No
PCT/US 99/27726

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G05B13/04

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 463 555 A (WARD DENNIS M ET AL) 31 October 1995 (1995-10-31) column 13, line 64 -column 33, line 15; figures 2-11	1-21
A	WO 98 50832 A (HONEYWELL INC) 12 November 1998 (1998-11-12) cited in the application abstract; figure 2	1,12,20
A	EP 0 756 219 A (WORTBERG JOHANNES PROF DR ;HAEUSSLER JOERG DR (DE)) 29 January 1997 (1997-01-29) abstract	1,12,20

[ ] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 March 2000	04/04/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Helot, H

Information on patent family members

International Application No
PCT/US 99/27726

Patent document cited in search report		Publication dat	Patent family member(s)		Publication date
US 5463555	A	31-10-1995	CA	2171846 A	06-04-1995
			EP	0721625 A	17-07-1996
			JP	9503325 T	31-03-1997
			WO	9509401 A	06-04-1995
WO 9850832	A	12-11-1998	AU	7286098 A	27-11-1998
EP 0756219	A	29-01-1997	DE	19514535 A	31-10-1996

THIS PAGE BLANK (USPTO)